

BIROn - Birkbeck Institutional Research Online

Tian, Z. and Zhang, H. and Chen, Y. and Zhang, Dell (2020) Unsupervised Hashing based on the Recovery of Subspace Structures. Pattern Recognition 103 , p. 107261. ISSN 0031-3203.

Downloaded from: <https://eprints.bbk.ac.uk/id/eprint/30968/>

Usage Guidelines:

Please refer to usage guidelines at <https://eprints.bbk.ac.uk/policies.html>
contact lib-eprints@bbk.ac.uk.

or alternatively

Unsupervised Hashing based on the Recovery of Subspace Structures

Zhibao Tian^a, Hui Zhang^{a,b}, Yong Chen^{c,*}, Dell Zhang^d

^a*School of Computer Science and Engineering, Beihang University, Beijing 100191, P.R. China*

^b*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, P.R. China*

^c*School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P.R. China*

^d*Birkbeck, University of London, London, the United Kingdom*

Abstract

Unsupervised semantic hashing should in principle keep the semantics among samples consistent with the intrinsic geometric structures of the dataset. In this paper, we propose a novel multiple stage unsupervised hashing method, named “*Unsupervised Hashing based on the Recovery of Subspace Structures*” (RSSH) for image retrieval. Specifically, we firstly adapt the Low-rank Representation (LRR) model into a new variant which treats the real-world data as samples drawn from a union of several low-rank subspaces. Then, the pairwise similarities are represented in a space-and-time saving manner based on the learned low-rank correlation matrix of the modified LRR. Next, the challenging discrete graph hashing is employed for binary hashing codes. Notably, we convert the original graph hashing model into an optimization-friendly formalization, which is addressed with efficient closed-form solutions for its subproblems. Finally, the devised linear hash functions are fast achieved for out-of-samples. Retrieval experiments on four image datasets testify the superiority of RSSH to several state-of-the-art hashing models. Besides, it’s worth mentioning that RSSH, a shallow model, significantly outperforms two recently proposed unsupervised deep hashing methods, which further confirms its effectiveness.

Keywords: Semantic Hashing, Subspace Learning, Low-rank Representation, Discrete Optimization

*Corresponding author: alphawolf.chen@gmail.com.

1. Introduction

Owing to the expense of low storage and fast computing, learning to hash has been widely accepted as an effective and efficient technique for large-scale image retrieval [1, 2, 3]. For example, given a collection of 1 billion images, if we adopt the traditional real-valued vector sized by 64×1 for each image, then the total memory should be “64 Dimensions/image \times 8 Bytes/Dimension $\times 10^9$ images = 512GB”; Nevertheless, if we leverage binary codes, then “64 Dimensions/image \times 1 bit/Dimension $\times 10^9$ images = 8GB” is sufficient. Besides, the hardware-level XOR operations are significantly faster than the floating-point calculations, which would greatly benefit image search engines. Such advantages have inspired many researchers to develop various hashing models, but there still exist great possibilities to devise more practical solutions for real-world applications.

Generally, hashing methods can be roughly divided into two categories: the supervised and unsupervised. The supervised hashing approaches such as Supervised Hashing with Kernels (KSH) [4], Fast Supervised Hashing with Decision Trees (FastH) [5], Supervised Discrete Hashing (SDH) [6, 7], Column Sampling based Discrete Supervised Hashing (COSDISH) [8], Fast Scalable Supervised Hashing (FSSH) [9], Semantic-Aware Discrete Hashing (SADIH) [10] and Deep Hashing based on Classification and Quantization errors (DHCQ) [11] usually yield competitive retrieval performance by utilizing the semantic labels, which in fact are quite difficult to obtain in many real-world scenarios, where we can only conduct unsupervised hashing. Besides, the performance of the unsupervised hashing methods is far from practical compared with that of the supervised ones. On top of these two reasons, we mainly focus on the unsupervised hashing in this paper.

Many popular unsupervised hashing models such as Spectral Hashing (SH) [12], Binary Reconstructive Embedding (BRE) [13], Anchor Graph Hashing (AGH) [14], Discrete Graph Hashing (DGH) [15], and Scalable Graph Hashing (SGH) [16], capture the complex structures of data by preserving the neighborhood similarities, which is quite effective because the graph-based learning could well unveil various nonlinear structures [17]. However, they also have limitations because of their modeling on the

local instead of the global structures. Low-rank Representation (LRR) [18, 19, 20] recognizes that data samples are approximately drawn from a mixture of several low-rank subspaces and then robustly recovers them by a rank minimization model. In this paper, we adapt the LRR method in a space-and-time saving fashion for the global structures preserved semantics. To the best of our knowledge, this hasn’t ever been tried in hashing.

To maintain the semantics harvested from the recovery of subspace structures by the modified LRR, we adopt the discrete graph hashing formalization for binary codes. Note that although this has been proposed in SH [12] and DGH [15], yet the existing solutions are usually underperformed for image retrieval due to either the continuous relaxations of binary constraints or not that effective optimizations. In this paper, we present a different method to solve the complex binary-constrained optimization with the help of two surrogate variables.

Briefly, we propose an unsupervised hashing method titled “*Unsupervised Hashing based on the Recovery of Subspace Structures*” (RSSH for short), which firstly learns the semantics by recovering the subspace structures of data, and then settles the strongly constrained semantic hashing by leveraging auxiliary variables. The main contributions can be listed as follows:

- RSSH adapts the LRR model into a new variant, based on which the learned correlation matrix could be designed into a space-and-time saving formula for data semantics.
- To tackle the discrete graph hashing, RSSH presents a new learning method, i.e., transforms the original optimization problem into three subproblems by means of surrogate variables, and most importantly each subproblem is addressed with a closed-form solution, which makes the whole hashing learning converge within dozens of iterations.
- Experiments on four datasets demonstrate the advantages of RSSH over several state-of-the-art unsupervised hashing models. Notably, even compared with the recently proposed unsupervised deep hashing methods (i.e., SADH [21] and DeepBit [22]), RSSH still outperforms them by an obvious gap.

2. Related Work

Our proposed method is an unsupervised semantic hashing, whose semantics between samples are calculated based on the recovery of subspace structures of the data. Hence, we relate our work to *Unsupervised Hashing* and *Subspace Learning*.

2.1. Unsupervised Hashing

Perhaps, the simplest unsupervised hashing methods belong to the data-independent Locality Sensitivity Hashing (LSH) and its variants [23, 24], which just employ a random matrix for projections and then binarize them according to a preset threshold. Although such approaches are extremely fast efficient, yet they usually exhibit very limited retrieval performance compared with the data-dependent competitors. Principal Component Analysis based Hashing (PCAH) [25] seeks an orthogonal subspace in which the variances of projected data samples could be maximized. However, the direct binarizations for the continuous embeddings in PCAH would probably lead to sub-optimal hashing codes due to the mismatch between the orthogonal subspace and the binary Hamming space. To address such puzzle, Iterative Quantization (ITQ) [25] finds a rotation of zero-centered data aiming at aligning the continuous space to the vertices of a zero-centered binary hypercube. Note that methods like PCAH and ITQ mainly concentrate on the dataset’s global structures with linear projections which would underperform in nonlinear datasets.

Graph-based hashing models such as Spectral Hashing (SH) [12], Binary Reconstructive Embedding (BRE) [13], Anchor Graph Hashing (AGH) [14], Discrete Graph Hashing (DGH) [15], and Scalable Graph Hashing (SGH) [16], all learn the binary hashing codes under the guidance of pairwise similarities between data samples. Note that the neighborhood semantics in the graph-based learning can preserve the local structures hidden in complex datasets. This is a popular learning trick to model the overall nonlinear structures by keeping the local structures. While deep learning based hashing methods, such as DH [26], DeepBit [22], UH-BDNN [27], SSDH [28] and SADH [21], capture the nonlinear relationship of data samples by the hierarchical nonlinear transformations. It’s commonly recognized that deep hashing models would

require an enormous volume of training data and large quantities of computing resources to learn the numerous network parameters for obtaining competitive retrieval results.

In this paper, we argue that the underlying complex structures in datasets can be modeled as a union of several low-rank subspaces [18, 29, 30], based on which an unsupervised hashing approach is further developed for image retrieval. Note that the subspace structures preserved similarities between samples are distinctive from the widely-used neighborhood similarities mentioned above. In addition, we devise a new learning method to solve the strictly binary-constrained graph hashing proposed in SH [12], AGH [14] and DGH [15] for obtaining the hashing codes.

2.2. Subspace Learning

Traditional linear learning models such as Principal Component Analysis (PCA) [31], Linear Discriminative Analysis (LDA) [32], and Non-negative Matrix Factorization (NMF) [33] have been widely used to embed each high-dimensional data sample into a compact low-dimensional vector. These methods differ in their noise assumptions, the use of prior information, and the underlying statistical models, but they all can recover the linear structure with only one low-rank subspace. To generalize it for capturing more complex structures, Generalized Principal Component Analysis (GPCA) [34] models the complex data as samples drawn from a union of multiple subspaces and formalizes it as a constrained nonlinear least square problem, which, however, is sensitive to noises/outliers and can be quite time-consuming. Sparse Subspace Clustering (SSC) [35] utilizes the sparse compressed sensing techniques to unveil the mixture of low-rank subspaces, but it probably underperforms due to the inaccurate capture of the global structures when the data is grossly corrupted. Low-rank Representation (LRR) [18, 19] could robustly recover the global structures by finding the lowest-rank representation of all data jointly. Note that the learned correlation matrix can be further processed into the semantic similarities among data samples.

However, the high space and computational complexities of the similarity matrix (sized by $n \times n$, n is the number of data samples in the dataset) learned in LRR would hinder its large-scale real-world applications. In this paper, we propose a new variant of the LRR model that could not only recover the overall subspace structures but also

encode the similarity matrix as the multiplications of low-rank matrices, which would reduce the space and computational complexities in the subsequent hashing learning.

3. Problem Statement

For the sake of formal presentations, we firstly introduce the notations adopted in this paper. Boldface lowercase letters like \mathbf{a} denote vectors. Boldface uppercase letters like \mathbf{A} denote matrices, and $\mathbf{A}_{i,j}$ represents the element at the i^{th} row and the j^{th} column of \mathbf{A} ; besides, \mathbf{A}_{i*} and \mathbf{A}_{*j} corresponds to the i^{th} row and the j^{th} column of \mathbf{A} respectively. \mathbf{O} marks a matrix with all 0-elements. Considering that n is a positive integer, \mathbf{I}_n denotes an $n \times n$ identity matrix, $\mathbf{1}_n$ represents an $n \times 1$ vector with all 1-elements, and $\mathbf{0}_n$ marks an $n \times 1$ vector with all 0-elements. \mathbf{A}^T means the transpose of \mathbf{A} , and \mathbf{A}^{-1} indicates the inverse of \mathbf{A} . $\max(\cdot)$ and $\min(\cdot)$ are functions which return the biggest and smallest element of a matrix or vector. $\text{sgn}(\cdot)$ is the element-wise sign function which returns 1 if the element is positive or -1 otherwise. Besides, $|\cdot|$ represents an element-wise absolute operator, $\text{rank}(\cdot)$ expresses the rank of the input matrix, \odot denotes the element-wise product, and $\|\mathbf{A}\|_F$ marks the Frobenius norm of \mathbf{A} .

Given a set of training samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, the goal is to learn their compact binary codes $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n]^T \in \{+1, -1\}^{n \times q}$, which should well preserve the semantics between them. Note that n is the number of samples in the dataset, d marks the dimension of the vector for each sample, and q represents the length of the learned binary vector for each sample. By convention, the semantics in the embedded binary space are usually measured by the Hamming distance. In particular, if two samples are similar in the original space, then the corresponding embedded vectors should be within a small Hamming distance; and vice versa. Besides, to vectorize the out-of-samples, we further learn q hash functions $H(\cdot) = [h_1(\cdot), h_2(\cdot), \dots, h_q(\cdot)]^T$ which could map each new data sample \mathbf{x}_i into q -bit binary codes, i.e., $\mathbf{b}_i = H(\mathbf{x}_i) \in \{+1, -1\}^q$. With the learned binary codes for existing databases and hashing functions for out-of-samples, relevant results can be efficiently retrieved for a new query via fast semantic computing.

4. The Method

The proposed method is an unsupervised multi-stage hashing model (Fig. 1) including: (i) low-rank representation, (ii) similarity matrix construction, (iii) semantics preserved hashing, (iv) learning hashing functions, and (v) retrieval application, whose specific details are elaborated as follows.

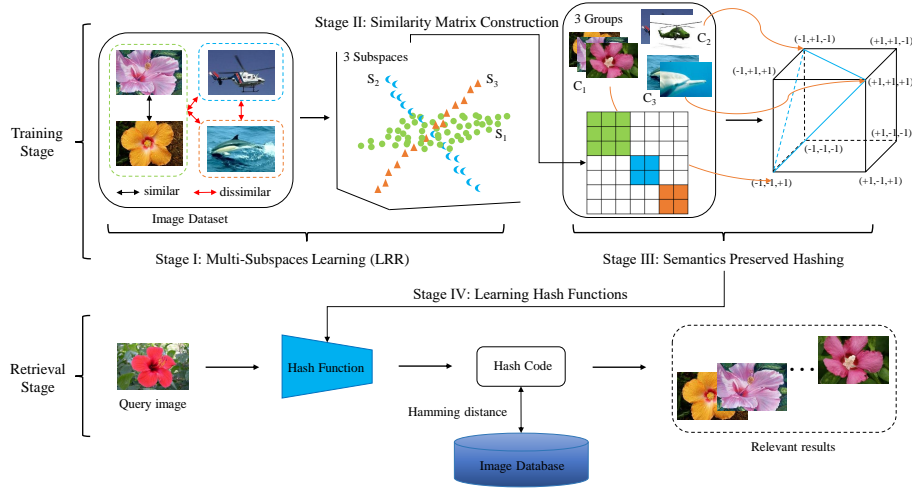


Figure 1: The overview framework of RSSH: multi-stage learning and its retrieval application.

4.1. Low-Rank Representation

Given a real-world image dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$, it usually holds complex structures which are reasonably modeled as a mixture of several low-rank subspaces. Therefore, to recover such subspace structures from \mathbf{X} , the Low-rank Representation (LRR) [19, 36] model can be employed as follow:

$$\min_{\mathbf{Z}, \mathbf{E}} \text{rank}(\mathbf{Z}) + \alpha \|\mathbf{E}\|_F^2, \text{ s.t. } \mathbf{X} = \mathbf{Z}\mathbf{X} + \mathbf{E}, \quad (1)$$

where the matrix $\mathbf{Z} \in \mathbb{R}^{n \times n}$ can be treated as the correlation matrix among data samples; the matrix \mathbf{E} represents the gaussian noises (here we adopt the popular gaussian distribution, instead of the $L_{2,1}$ or L_1 distributions employed in [19], to model the complex unknown noises under the background of big data); and the positive parameter

α is utilized to balance the importance between $\text{rank}(\mathbf{Z})$ and $\|\mathbf{E}\|_F^2$. Note that minimizing the objective function in (1) aims to keep as much information as possible and meanwhile seek the compact/low-rank subspace structures of the dataset.

Based on the correlation matrix \mathbf{Z} , the similarity matrix \mathbf{S} among data samples are calculated as $\mathbf{S} = |\mathbf{Z}| + |\mathbf{Z}^T|$ in Refs. [35, 18]. However, \mathbf{S} in this form needs high space and time complexity which will hinder the efficiency of subsequent optimizations. Hence, we develop a new formula¹, i.e., $\mathbf{S} = \mathbf{Z} \odot \mathbf{Z} + \mathbf{Z}^T \odot \mathbf{Z}^T$ to compute the similarity matrix. Although it looks much more complex than the original computational formula, yet it could be transformed into a much more efficient form in Eq. (8), i.e., time-efficiency low-rank matrix multiplications. To achieve this, we can replace the low-rank matrix \mathbf{Z} with the multiplications of two auxiliary matrices \mathbf{U} and \mathbf{V} (i.e., $\mathbf{Z} = \mathbf{UV}^T$), and then approximately convert the optimization (1) into:

$$\min_{\mathbf{U}, \mathbf{V}, \mathbf{E}} \|\mathbf{E}\|_F^2, \text{ s.t. } \mathbf{X} = \mathbf{UV}^T \mathbf{X} + \mathbf{E}, \quad (2)$$

where $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{n \times r}$. Note that r is a parameter that can control the rank of \mathbf{Z} which is based on the fact that $\text{rank}(\mathbf{Z}) = \text{rank}(\mathbf{UV}^T) \leq \min(\text{rank}(\mathbf{U}), \text{rank}(\mathbf{V})) \leq r$ (generally $r \ll n$). On top of this, the optimization (2) is further simplified as:

$$\min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T \mathbf{X}\|_F^2, \quad (3)$$

which is an unconstrained optimization problem.

4.2. Similarity Matrix Construction

Before computing the above defined similarity matrix \mathbf{S} , a theorem is introduced first.

Theorem 1. *Given two matrices $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{n \times r}$, set $\tilde{\mathbf{U}} = \mathbf{U} \odot \mathbf{U}$ and $\tilde{\mathbf{V}} = \mathbf{V} \odot \mathbf{V}$; besides, construct matrices $\bar{\mathbf{U}} \in \mathbb{R}^{n \times l}$ and $\bar{\mathbf{V}} \in \mathbb{R}^{n \times l}$ ($l = r \times (r-1)/2$), where $\bar{\mathbf{U}}_{k,t} = \mathbf{U}_{k,i} \times \mathbf{U}_{k,j}$ and $\bar{\mathbf{V}}_{k,t} = \mathbf{V}_{k,i} \times \mathbf{V}_{k,j}$. Note that $k = 1, 2, \dots, n$; $i, j = 1, 2, \dots, r$ and $i < j$; $t = \sum_{g=1}^{i-1} (r-g) + j - i$. Then, there exists:*

$$\mathbf{S}_h \equiv (\mathbf{UV}^T) \odot (\mathbf{UV}^T) = \tilde{\mathbf{U}}\tilde{\mathbf{V}}^T + 2\bar{\mathbf{U}}\bar{\mathbf{V}}^T. \quad (4)$$

¹The difference from the original formula is that the square instead of the absolute value of each element is leveraged for calculations.

Proof. For $1 \leq k, g \leq n$, there holds:

$$\begin{aligned}
([\mathbf{UV}^T]_{k,g})^2 &= [(\mathbf{UV}^T) \odot (\mathbf{UV}^T)]_{k,g} \\
&= \left(\sum_{w=1}^r \mathbf{U}_{k,w} \mathbf{V}_{g,w} \right)^2 \\
&= \sum_{w=1}^r (\mathbf{U}_{k,w} \mathbf{V}_{g,w})^2 + 2 \sum_{1 \leq w < h \leq r} (\mathbf{U}_{k,w} \mathbf{V}_{g,w})(\mathbf{U}_{k,h} \mathbf{V}_{g,h}) \\
&= \sum_{w=1}^r (\mathbf{U}_{k,w})^2 (\mathbf{V}_{g,w})^2 + 2 \sum_{1 \leq w < h \leq r} (\mathbf{U}_{k,w} \mathbf{U}_{k,h})(\mathbf{V}_{g,w} \mathbf{V}_{g,h}) \\
&= [(\mathbf{U} \odot \mathbf{U})(\mathbf{V} \odot \mathbf{V})^T]_{k,g} + 2 \sum_{t=1}^l \bar{\mathbf{U}}_{k,t} \bar{\mathbf{V}}_{g,t} \\
&= [\tilde{\mathbf{U}} \tilde{\mathbf{V}}^T]_{k,g} + 2[\bar{\mathbf{U}} \bar{\mathbf{V}}^T]_{k,g},
\end{aligned}$$

which means that:

$$(\mathbf{UV}^T) \odot (\mathbf{UV}^T) = \tilde{\mathbf{U}} \tilde{\mathbf{V}}^T + 2\bar{\mathbf{U}} \bar{\mathbf{V}}^T;$$

thus this theorem is proved. \square

In what follows, we scale each element of \mathbf{S}_h to the domain $[0, 1]$ by:

$$\mathbf{S}_h \leftarrow \frac{\mathbf{S}_h - \min(\mathbf{S}_h)}{\max(\mathbf{S}_h) - \min(\mathbf{S}_h)}. \quad (5)$$

In practice, we could observe that $\min(\mathbf{S}_h)$ is so close to 0 and $\max(\mathbf{S}_h)$ is always among the diagonal elements of \mathbf{S}_h , so we use 0 to replace $\min(\mathbf{S}_h)$ and $M = \max(\text{diag}(\mathbf{S}_h))$ to replace $\max(\mathbf{S}_h)$, where $\text{diag}(\mathbf{S}_h)$ denotes the vector composed of the diagonal elements of \mathbf{S}_h . With the help of Eq. (4), the computational complexity of $\max(\mathbf{S}_h)$ is shortened from $O(n^2)$ to $O(n)$. Then we reduce Eq. (5) to:

$$\mathbf{S}_h \leftarrow \frac{1}{M} \mathbf{S}_h. \quad (6)$$

To further prepare the similarity matrix \mathbf{S} better for the sequel semantic hashing, we then center its each element to the value ranging from -1 to $+1$ as below:

$$\mathbf{S} \leftarrow \mathbf{S} - \mathbf{1}\mathbf{1}^T = \mathbf{S}_h + \mathbf{S}_h^T - \mathbf{1}\mathbf{1}^T. \quad (7)$$

According to Eqs. (4), (6) and (7), we finally construct \mathbf{S} as the sum of several products of low-rank matrices:

$$\mathbf{S} = \frac{1}{M}(\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T + 2\bar{\mathbf{U}}\bar{\mathbf{V}}^T + \tilde{\mathbf{V}}\tilde{\mathbf{U}}^T + 2\bar{\mathbf{V}}\bar{\mathbf{U}}^T) - \mathbf{1}\mathbf{1}^T. \quad (8)$$

Note that \mathbf{S} is not actually computed, when using \mathbf{S} in the following equations, we replace \mathbf{S} with Eq. (8).

4.3. Semantics Preserved Hashing

Considering a pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$ and their binary codes $(\mathbf{b}_i, \mathbf{b}_j)$, the similarity \mathbf{S}_{ij} in the original data space should be as close as possible to their semantic² $\frac{1}{q}\mathbf{b}_i^T\mathbf{b}_j$ in the Hamming space. In this way, we can preserve the semantics among data samples from the original to the embedded space. Formally, we can build the following graph hashing model:

$$\min_{\mathbf{B}} \sum_{ij} (q\mathbf{S}_{ij} - \mathbf{b}_i^T\mathbf{b}_j)^2 = \|\mathbf{qS} - \mathbf{B}\mathbf{B}^T\|_F^2 \quad (9)$$

$$\text{s.t. } \mathbf{B} \in \{+1, -1\}^{n \times q}, \mathbf{B}^T\mathbf{B} = n\mathbf{I}_q, \mathbf{B}^T\mathbf{1}_n = \mathbf{0}_q,$$

where the constraint $\mathbf{B}^T\mathbf{B} = n\mathbf{I}_q$ makes the q bits mutually uncorrelated, and the constraint $\mathbf{B}^T\mathbf{1}_n = \mathbf{0}_q$ is adopted to make half of the bits $-1/+1$, which together maximize the coding capabilities given fixed code length.

Obviously, the problem (9) is a discrete optimization with complex constraints which is hard to solve. Traditional strategies are usually to relax the discrete constraints to the continuous ones for much easier solutions (such as SH [12], BRE [13], and AGH [14]); however, as the number of bits becomes larger, the quantization errors will increase a lot, which would lead to suboptimal solutions for image retrieval [37]. To tackle such limitations, a few recent attempts such as KSH [4], DGH [15], SDH [6], COSDISH [8], and FSSH [9], using discrete optimization techniques, have shown fast learning and better performance.

²The semantic between \mathbf{b}_i and \mathbf{b}_j can be calculated by their cosine value, i.e., $\frac{1}{q}\mathbf{b}_i^T\mathbf{b}_j \in [-1, +1]$, which is consistent to \mathbf{S}_{ij} at scale.

Inspired by such works, we introduce two auxiliary variables \mathbf{A} and \mathbf{G} , and then convert the problem (9) into an equivalent one:

$$\begin{aligned} & \min_{\mathbf{B}, \mathbf{A}, \mathbf{G}} \|q\mathbf{S} - \mathbf{A}\mathbf{G}^T\|_F^2 \\ & \text{s.t.} \begin{cases} \mathbf{A} = \mathbf{B}, \mathbf{G} = \mathbf{B}; \\ \mathbf{B} \in \{+1, -1\}^{n \times q}, \mathbf{G} \in \mathbb{R}^{n \times q}; \\ \mathbf{A} \in \mathbb{R}^{n \times q}, \mathbf{A}^T \mathbf{A} = n\mathbf{I}_q, \mathbf{A}^T \mathbf{1}_n = \mathbf{0}_q. \end{cases}, \end{aligned} \quad (10)$$

where \mathbf{A} and \mathbf{G} can be viewed as the continuous surrogates of the binary matrix \mathbf{B} . In what follows, we could further relax the complex optimization (10) by dropping the constraints $\mathbf{A} = \mathbf{B}$ and $\mathbf{G} = \mathbf{B}$ as below:

$$\begin{aligned} & \min_{\mathbf{B}, \mathbf{A}, \mathbf{G}} \|q\mathbf{S} - \mathbf{A}\mathbf{G}^T\|_F^2 + \beta \|\mathbf{A} - \mathbf{B}\|_F^2 + \lambda \|\mathbf{G} - \mathbf{B}\|_F^2 \\ & \text{s.t.} \begin{cases} \mathbf{B} \in \{+1, -1\}^{n \times q}, \mathbf{G} \in \mathbb{R}^{n \times q}; \\ \mathbf{A} \in \mathbb{R}^{n \times q}, \mathbf{A}^T \mathbf{A} = n\mathbf{I}_q, \mathbf{A}^T \mathbf{1}_n = \mathbf{0}_q. \end{cases}, \end{aligned} \quad (11)$$

where β and λ are two non-negative hyper-parameters to adjust the closeness between \mathbf{B} and \mathbf{A}/\mathbf{G} respectively. In practice, it's common to set them to moderate values for real-world applications.

4.4. Learning Hash Functions

For fast codings in image retrieval, we design the hash functions in the following:

$$H(\mathbf{X}) = \phi(\mathbf{X})\mathbf{P}, \quad (12)$$

where $\phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_n)]^T$ and $\mathbf{P} \in \mathbb{R}^{m \times q}$ is the projection matrix which aims to transform $\phi(\mathbf{X})$ to the binary codes \mathbf{B} . It should be noted that $\phi(\mathbf{x})$ is an m -dimensional vector obtained by the **RBF** kernel mapping, i.e., $\phi(\mathbf{x}) = [\exp(-\|\mathbf{x} - \mathbf{a}_1\|^2/(2\sigma^2)), \dots, \exp(-\|\mathbf{x} - \mathbf{a}_m\|^2/(2\sigma^2))]$, where $\{\mathbf{a}_i\}_1^m$ are the randomly selected m anchor samples from the training dataset and σ is the kernel width. These two hyper-parameters would be tuned to the competitive settings via experiments. In what follows, to learn the hash functions, an optimization problem is modeled:

$$\min_{\mathbf{P}} \|\phi(\mathbf{X})\mathbf{P} - \mathbf{B}\|_F^2 + \eta \|\mathbf{P}\|_F^2, \quad (13)$$

where η is a smooth factor, and the optimal \mathbf{P} can be calculated with:

$$\mathbf{P} = (\phi(\mathbf{X})^T \phi(\mathbf{X}) + \eta \mathbf{I})^{-1} \phi(\mathbf{X})^T \mathbf{B}. \quad (14)$$

Out-of-Sample Extensions: For new queries denoted by \mathbf{X}_{new} , their hash codes could be achieved via:

$$\mathbf{B}_{new} = \text{sgn}(\phi(\mathbf{X}_{new})\mathbf{P}), \quad (15)$$

which can be executed in parallel and thus fast-efficient.

Since images could be quickly embedded in hamming space with the help of Eq. (15), the hashing-based retrieval system then rapidly returns the top- N (e.g., $N = 100$) relevant results based on the hamming distances between the given image query and the candidates in image database (illustrated in Fig. 1).

5. Optimization

5.1. Solution to problem (3)

Regarding the optimization problem (3), we could update \mathbf{U} and \mathbf{V} iteratively with the other fixed until convergence.

U Step. With \mathbf{V} fixed, the objective function of \mathbf{U} is given by:

$$\min_{\mathbf{U}} \mathcal{O}(\mathbf{U}) = \|\mathbf{X} - \mathbf{U}\mathbf{V}^T \mathbf{X}\|_F^2. \quad (16)$$

Unfold $\mathcal{O}(\mathbf{U})$ as follows:

$$\begin{aligned} \mathcal{O}(\mathbf{U}) &= \text{tr}((\mathbf{X} - \mathbf{U}\mathbf{V}^T \mathbf{X})(\mathbf{X}^T - \mathbf{X}^T \mathbf{V} \mathbf{U}^T)) \\ &= \text{tr}(\mathbf{X}\mathbf{X}^T - 2\mathbf{X}\mathbf{X}^T \mathbf{V} \mathbf{U}^T + \mathbf{U}\mathbf{V}^T \mathbf{X}\mathbf{X}^T \mathbf{V} \mathbf{U}^T) \\ &\propto \text{tr}(\mathbf{U}\mathbf{V}^T \mathbf{X}\mathbf{X}^T \mathbf{V} \mathbf{U}^T - 2\mathbf{X}\mathbf{X}^T \mathbf{V} \mathbf{U}^T). \end{aligned} \quad (17)$$

Calculate the derivative of \mathbf{U} and then set it to \mathbf{O} :

$$\frac{\partial \mathcal{O}(\mathbf{U})}{\partial \mathbf{U}} = 2\mathbf{U}\mathbf{V}^T \mathbf{X}\mathbf{X}^T \mathbf{V} - 2\mathbf{X}\mathbf{X}^T \mathbf{V} = \mathbf{O}, \quad (18)$$

based on which the closed solution is written as:

$$\mathbf{U} = \mathbf{X}\mathbf{X}^T \mathbf{V} (\mathbf{V}^T \mathbf{X}\mathbf{X}^T \mathbf{V})^{-1}. \quad (19)$$

Algorithm 1: Low-rank Representation

Input: Data matrix \mathbf{X} , the parameter r , and the maximum number of iterations $maxIter$.

Output: Matrices \mathbf{U} and \mathbf{V} .

- 1 Set $t = 1$ and randomly initialize $\mathbf{U}^{(0)}$ and $\mathbf{V}^{(0)}$;
 - 2 **while** $t < maxIter$ **do**
 - 3 Compute $\mathbf{U}^{(t)}$ according to Eq. (19);
 - 4 Compute $\mathbf{V}^{(t)}$ according to Eq. (21);
 - 5 $t = t + 1$;
 - 6 **end**
 - 7 Return $\mathbf{U}^{(t)}$ and $\mathbf{V}^{(t)}$.
-

V Step. With \mathbf{U} fixed, the optimization problem w.r.t. \mathbf{V} is simplified as:

$$\min_{\mathbf{V}} \mathcal{O}(\mathbf{V}) = \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\mathbf{X}\|_F^2, \quad (20)$$

whose solution can be easily obtained:

$$\mathbf{V} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}. \quad (21)$$

Based on the above \mathbf{U} and \mathbf{V} steps, we summarize the solution to problem (3) in Algorithm 1. Here, it's worth mentioning that the Eq. (19) and Eq. (21) might be irreversible in theory; thus, a small smooth item is usually added to avoid irreversibility in practice [38]. That is, $\mathbf{U} = \mathbf{X}\mathbf{X}^T\mathbf{V}(\mathbf{V}^T\mathbf{X}\mathbf{X}^T\mathbf{V} + \delta\mathbf{I})^{-1}$ and $\mathbf{V} = \mathbf{U}(\mathbf{U}^T\mathbf{U} + \delta\mathbf{I})^{-1}$, where δ is set to 1e-6 in our paper.

5.2. Solution to problem (11)

To solve the problem (11), we present an iterative optimization process, in which each iteration contains three steps, i.e., **A Step**, **G Step** and **B Step**.

A Step. With \mathbf{G} and \mathbf{B} fixed, the problem (11) is transformed into:

$$\begin{aligned} \min_{\mathbf{A}} \mathcal{O}(\mathbf{A}) &= \|q\mathbf{S} - \mathbf{A}\mathbf{G}^T\|_F^2 + \beta\|\mathbf{A} - \mathbf{B}\|_F^2 \\ \text{s.t. } \mathbf{A} &\in \mathbb{R}^{n \times q}, \mathbf{A}^T\mathbf{A} = n\mathbf{I}_q, \mathbf{A}^T\mathbf{1}_n = \mathbf{0}_q. \end{aligned} \quad (22)$$

Unfold Eq. (22) and we can arrive at:

$$\begin{aligned}\mathcal{O}(\mathbf{A}) &= \text{tr}(q^2 \mathbf{S} \mathbf{S}^T - 2q \mathbf{S}^T \mathbf{A} \mathbf{G}^T + \mathbf{A} \mathbf{G}^T \mathbf{G} \mathbf{A}^T) \\ &\quad + \beta \text{tr}(\mathbf{A} \mathbf{A}^T - 2\mathbf{A} \mathbf{B}^T + \mathbf{B} \mathbf{B}^T) \\ &\propto -\text{tr}((q \mathbf{S} \mathbf{G} + \beta \mathbf{B})^T \mathbf{A}),\end{aligned}\tag{23}$$

based on which the problem (22) is equivalent to:

$$\begin{aligned}\max_{\mathbf{A}} \quad & \text{tr}(\mathbf{C}^T \mathbf{A}) \\ \text{s.t. } \quad & \mathbf{A} \in \mathbb{R}^{n \times q}, \mathbf{A}^T \mathbf{A} = n \mathbf{I}_q, \mathbf{A}^T \mathbf{1}_n = \mathbf{0}_q,\end{aligned}\tag{24}$$

where $\mathbf{C} = q \mathbf{S} \mathbf{G} + \beta \mathbf{B}$.

Set the centering matrix $\mathbf{J} = \mathbf{I}_n - \frac{1}{n} \mathbf{1} \mathbf{1}^T$ and then do singular value decomposition of $\mathbf{J} \mathbf{C}$ as $\mathbf{J} \mathbf{C} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_{k=1}^{q'} \sigma_k \mathbf{u}_k \mathbf{v}_k^T$, where $q' \leq q$ is the rank of $\mathbf{J} \mathbf{C}$, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{q'}$ are the positive singular values, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{q'}]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{q'}]$. Next, by employing the Gram-Schmidt process, we can obtain matrices $\bar{\mathbf{U}} \in \mathbb{R}^{n \times (q-q')}$ and $\bar{\mathbf{V}} \in \mathbb{R}^{q \times (q-q')}$ such that $\bar{\mathbf{U}}^T \bar{\mathbf{U}} = \mathbf{I}_{q-q'}$, $[\mathbf{U}, \bar{\mathbf{U}}]^T \bar{\mathbf{U}} = \mathbf{O}$ and $\bar{\mathbf{V}}^T \bar{\mathbf{V}} = \mathbf{I}_{q-q'}$, $\mathbf{V}^T \bar{\mathbf{V}} = \mathbf{O}$. Here please note that if $q' = q$, then $\bar{\mathbf{U}}$ and $\bar{\mathbf{V}}$ will be nothing. In what follows, to solve the optimization (24), we could borrow the following theorem:

Theorem 2. $\mathbf{A} = \sqrt{n}[\mathbf{U}, \bar{\mathbf{U}}][\mathbf{V}, \bar{\mathbf{V}}]^T$ is the optimal solution of the maximization problem (24).

Proof. please refer to Ref. [15]. □

G Step. When \mathbf{A} and \mathbf{B} are fixed, the problem (11) is re-written as:

$$\min_{\mathbf{G}} \mathcal{O}(\mathbf{G}) = \|q \mathbf{S} - \mathbf{A} \mathbf{G}^T\|_F^2 + \lambda \|\mathbf{G} - \mathbf{B}\|_F^2,\tag{25}$$

whose closed solution can be easily achieved:

$$\mathbf{G} = \frac{1}{n + \lambda} (q \mathbf{S}^T \mathbf{A} + \lambda \mathbf{B}).\tag{26}$$

B Step. Keeping \mathbf{A} and \mathbf{G} fixed, we can convert the problem (11) as below:

$$\min_{\mathbf{B}} \mathcal{O}(\mathbf{B}) = \beta \|\mathbf{A} - \mathbf{B}\|_F^2 + \lambda \|\mathbf{G} - \mathbf{B}\|_F^2\tag{27}$$

Algorithm 2: Semantics Preserved Hashing

Input: Similarity matrix \mathbf{S} , length of hash codes q , hyper-parameters β and λ ,
the maximum number of iterations $maxIter$.

Output: Binary codes \mathbf{B} .

```
1 Set  $t = 1$  and randomly initialize  $\mathbf{B}^{(0)}$ ,  $\mathbf{G}^{(0)}$  and  $\mathbf{A}^{(0)}$ ;  
2 while  $t < maxIter$  do  
3   Compute  $\mathbf{A}^{(t)}$  according to Theorem 2;  
4   Compute  $\mathbf{G}^{(t)}$  according to Eq. (26);  
5   Compute  $\mathbf{B}^{(t)}$  according to Eq. (30);  
6    $t = t + 1$ ;  
7 end  
8 Return  $\mathbf{B}^{(t)}$ ,  $\mathbf{G}^{(t)}$  and  $\mathbf{A}^{(t)}$ .
```

$$s.t. \mathbf{B} \in \{-1, +1\}^{n \times q}.$$

The following shows the equivalent form of $\mathcal{O}(\mathbf{B})$:

$$\mathcal{O}(\mathbf{B}) \propto -tr(\mathbf{B}^T(\beta\mathbf{A} + \lambda\mathbf{G})). \quad (28)$$

Therefore, the optimization (27) is equivalent to:

$$\max_{\mathbf{B}} tr(\mathbf{B}^T(\beta\mathbf{A} + \lambda\mathbf{G})) \quad s.t. \mathbf{B} \in \{+1, -1\}^{n \times q}, \quad (29)$$

whose solution is summarized in:

$$\mathbf{B} = \text{sgn}(\beta\mathbf{A} + \lambda\mathbf{G}). \quad (30)$$

The algorithm for learning hash codes is built on the \mathbf{A} , \mathbf{G} , \mathbf{B} steps and further concluded in Algorithm 2.

To sum up, our proposed method RSSH is a four-stage learning process, i.e., (i) learn the low rank representation according to the Algorithm 1; (ii) construct the similarity matrix according to Eq. (8); (iii) Obtain hash codes according to the Algorithm 2; (iv) learn the hash functions by Eq. (14), illustrated in Algorithm 3.

Algorithm 3: RSSH

Input: Data matrix \mathbf{X} , the maximum number of iterations for Algorithm 1 and Algorithm 2 respectively, the parameter r , length of hash codes q , hyper-parameters $\beta, \lambda, \eta, \sigma$ and m .

Output: Binary codes \mathbf{B} and hash functions \mathbf{P} .

- 1 Learn the low-rank representation matrices \mathbf{U} and \mathbf{V} according to Algorithm 1;
 - 2 Construct similarity matrix according to Eq. (8);
 - 3 Obtain hash codes according to Algorithm 2;
 - 4 Learn hash functions by Eq. (14);
 - 5 Return \mathbf{B} , \mathbf{A} , \mathbf{G} and \mathbf{P} .
-

5.3. Computational Complexity Analysis

The time spending of the whole Algorithm 3 is mainly concentrated on Algorithm 1, Algorithm 2 and the learning of hash functions.

For Algorithm 1, two closed-form solutions, i.e., Eq. (19) and Eq. (21), are derived for the corresponding two subproblems, whose computational complexity are $O(ndr + nr^2 + r^3)$ and $O(nr^2 + r^3)$ respectively for each iteration. Therefore, the whole computational complexity should be $O(n \times \max(d, r) \times r \times \maxIter)$, where \maxIter is 100.

For Algorithm 2, there are three subproblems. Regarding the \mathbf{A} -subproblem, the main step requiring intensive computation is the singular value decomposition for a matrix sized by $n \times q$, whose computational complexity is $O(nq^2)$. It is easy to validate that the other steps would need much less time expenditure. Regarding the \mathbf{G} -subproblem, the most time-consuming part is $\mathbf{S}^T \mathbf{A}$. In practice, we replace \mathbf{S} with the right side of Eq. (8), which reduces the computational complexity from $O(n^2q)$ to $O(nr^2q)$. Regarding the \mathbf{B} -subproblem, the computational complexity of Eq. (30) is $O(nq)$. Thus, the whole computational complexity should be $O(nr^2q \times \maxIter)$, where \maxIter is configured as 20 here.

For the closed-form solution to learn hash functions \mathbf{P} , the computational complexity of Eq. (14) is $O(nm^2 + m^3 + nmq)$, where m is the number of anchor samples randomly

selected from the training data.

Generally, for a large-scale dataset, r, q, m are usually much less than n , and thus the total computational complexity of the RSSH algorithm is linear to n (the number of training samples). Note that the whole computational process is mainly based on the matrix multiplications, which can be computed in parallel for accelerations and hence reasonably fast-efficient on big data.

6. Experiment

This section will narrates the significant advantages of our proposed method over several state-of-the-art unsupervised competitors (including two newly designed unsupervised deep hashing approaches) via a series of experiments.

6.1. Datasets

To evaluate the retrieval performances of various representative hashing methods, four widely-accepted image datasets are selected as below.

USPS³ is a handwritten digit database which contains 11,000 images of digits from “0” to “9” [39]. Specifically, each digit has 1,100 images with each in 16×16 pixels; therefore, a 256-dimensional vector is coded for representing each image.

Caltech256⁴ collects 30,607 images belonging to 256 categories [40]. Each image is encoded by a 1,024-dimensional CNN feature vector associated with one category label. Note that this processed dataset is directly downloaded from <https://github.com/willard-yuan/hashing-baseline-for-image-retrieval>.

Fashion-MNIST⁵ is similar to but a more challenging image dataset than the handwritten digit set **MNIST**⁶ [41]. It covers 70,000 images of different products from 10 categories. Each category holds 7,000 images with each one in $28 \times 28 = 784$ pixel resolution.

³<https://cs.nyu.edu/~roweis/data.html>

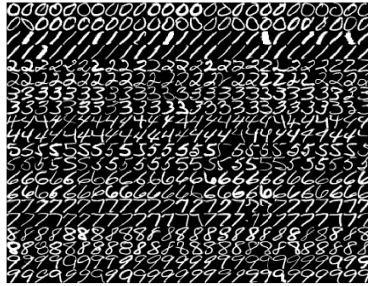
⁴http://www.vision.caltech.edu/Image_Datasets/Caltech256

⁵<https://github.com/zalandoresearch/fashion-mnist>

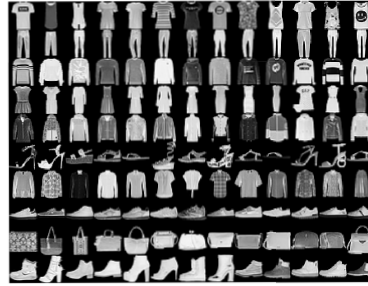
⁶<http://yann.lecun.com/exdb/mnist>

NUS-WIDE⁷ is a real-world web database originally containing 269,648 images each associated with multiple textual tags [42]. Following the protocol in Ref. [43], we use the images that cover the top 10 most frequent semantic concepts and finally obtain 186,577 images. Each image is converted into a 500-dimensional bag-of-visual-word features. Note that this dataset is a relatively larger and more challenging dataset for image retrieval.

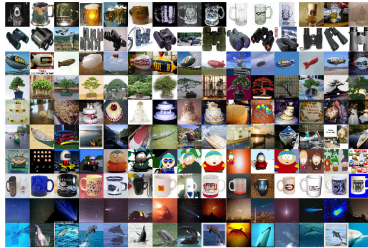
For all the above datasets **USPS**, **Caltech256**, **Fashion-MNIST** and **NUS-WIDE**, 1,000 samples are randomly chosen as the testing set and the remaining comes to the training set. **Fig. 2 illustrates some sample images from each dataset: USPS talks about handwritten digits; Fashion-MNIST depicts about clothes; Caltech256 and NUS-WIDE focus on colorful daily-life scenarios with various styles.**



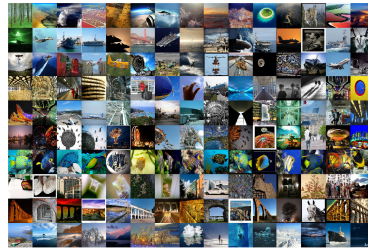
(a) USPS: sample images



(b) Fashion-MNIST: sample images



(c) Caltech256: sample images



(d) NUS-WIDE: sample images

Figure 2: Illustrations: sample images from **USPS**, **Caltech256**, **Fashion-MNIST** and **NUS-WIDE** datasets.

⁷<https://scholarbank.nus.edu.sg/handle/10635/40888>

6.2. Evaluation Metrics and Baseline Competitors

For all the retrieval experiments on the above datasets, we utilize several popular evaluation metrics [44, 45] such as Mean Average Precision (MAP), Precision@topN and Recall@topN, and PR-curves to assess the performance of various hashing competitors. Note that MAP and PR-curves evaluate the overall performance of the image retrieval systems, and Precision/Recall@topN measure the precision and recall at fixed levels of retrieved results.

When it comes to baseline methods for comparisons, we choose quite a few state-of-the-art unsupervised shallow methods: LSH⁸ [1], PCAH⁹ [25], ITQ¹⁰ [25], DGH [15], SGH¹¹ [16], SpH¹² [46], SH¹³ [12], and SELVE¹⁴ [47]. In addition, we also select two recently proposed deep hashing methods, i.e., DeepBit¹⁵ [22] and SADH¹⁶ [21]. The source codes of all the baselines except for DGH are publicly available online. Regarding DGH, we implemented it in strict accordance with the instructions of Ref. [15]. To ensure a fair comparison, the inputs of all methods are kept the same. With respect to the hyper-parameters, we tune them on different datasets for the best performances according to the corresponding papers’ proposals.

6.3. Experimental Setups

The RSSH’s parameters are elaborated here for the reproduction of the experimental results. Specifically, we set the maximum number of iterations as 100 and 20 for Algorithm 1 and Algorithm 2 respectively. In light of the parameter r which controls the rank of matrix \mathbf{S} , we tune it from 10 to 300 and finally configure it with 100 for all datasets. Regarding the other hyper-parameters β , λ and η , we

⁸<http://www.cad.zju.edu.cn/home/dengcai/Data/DSH.html>

⁹<https://github.com/willard-yuan/hashing-baseline-for-image-retrieval>

¹⁰<https://goo.gl/AGuu86>

¹¹<http://cs.nju.edu.cn/lwj/code/SGH.rar>

¹²http://sglab.kaist.ac.kr/projects/Spherical_Hashing/static/media/

[uploads/Spherical_Hashing_Src_Matlab.zip](#)

¹³<http://www.cs.huji.ac.il/~yweiss/SpectralHashing/sh.zip>

¹⁴<https://github.com/willard-yuan/hashing-baseline-for-image-retrieval>

¹⁵<https://github.com/kevinlin311tw/cvpr16-deepbit>

¹⁶<https://github.com/xuyan1115/Similarity-Adaptive-Deep-Hashing>

Table 1: The MAP scores and training time of different methods on **USPS**. Note: the best results are in bold and the second-best are underlined.

Methods	USPS				Training time (Seconds)			
	16bits	32bits	64bits	96bits	16bits	32bits	64bits	96bits
LSH	0.1865	0.2179	0.2803	0.3094	0.001	0.001	0.001	0.002
PCAH	0.1779	0.1900	0.2097	0.2282	0.028	0.037	0.040	0.049
ITQ	0.2814	0.2823	<u>0.3460</u>	0.3660	0.205	0.501	1.028	1.649
DGH	0.2556	0.2652	0.2823	0.3285	13.941	22.569	22.966	32.721
SGH	<u>0.3214</u>	<u>0.3308</u>	0.3395	0.3486	0.949	6.248	22.835	57.361
SpH	0.2516	0.3102	0.3300	0.3601	0.323	0.608	1.318	2.046
SH	0.2345	0.2353	0.2376	0.2439	0.040	0.055	0.083	0.103
SELVE	0.2959	0.3136	0.3305	<u>0.3978</u>	0.985	1.082	1.083	1.176
RSSH	0.4940	0.4990	0.5073	0.5243	8.848	8.933	9.488	10.533

empirically set each of them ranging from 0, 10^{-7} to 10^7 with the others fixed, and then plot their performance curves respectively (Fig. 5b-5d). In the end, we select one group of parameters under which all experiments can produce competitive results, i.e., $(\beta, \lambda, \eta) = (0.1, 10, 10^{-6})$. In terms of the **RBF** kernel mapping: $\phi(\mathbf{x}) = [\exp(-\|\mathbf{x} - \mathbf{a}_1\|^2 / (2\sigma^2)), \dots, \exp(-\|\mathbf{x} - \mathbf{a}_m\|^2 / (2\sigma^2))]$, we set the number of anchor points $m = 2,000$ and the kernel width $\sigma = 0.5$ for competitive performances. Note that the detailed parameter sensitivity analysis are further presented in Section 6.6.

6.4. Results

Table 1,2,3,4 show the MAP values of various competitors on four datasets with different bits from 16 to 96; Fig. 3 exhibits the precision and recall for the topN returned results with 64 bits, and Fig. 4 displays the PR curves on different datasets with 64 bits. It's worth mentioning that the experimental results (Precision/Recall@topN, PR curves) with other bits hold the similar trends as those with 64 bits in Figs 3&4. Thus, we here omit them for saving space.

Clearly, no matter what kind of measurements, i.e., MAP, PR-Curves, Precision@topN and Recall@topN, RSSH consistently outperforms all the baseline methods on the whole,

Table 2: The MAP scores and training time of different methods on **Caltech256**. Note: the best results are in bold and the second-best are underlined.

Methods	Caltech256				Training time (Seconds)			
	16bits	32bits	64bits	96bits	16bits	32bits	64bits	96bits
LSH	0.0354	0.0712	0.1321	0.1926	0.001	0.001	0.018	0.003
PCAH	0.0977	0.1590	0.2037	0.2176	0.474	0.496	0.532	0.575
ITQ	0.1445	0.2326	0.2998	0.3356	1.132	2.041	3.578	5.035
DGH	0.0884	0.1282	0.1606	0.1801	35.337	38.910	59.143	60.352
SGH	0.1331	0.2174	0.2906	0.3256	2.387	3.047	4.867	6.758
SpH	0.0693	0.1308	0.1858	0.2176	2.194	2.647	4.346	6.895
SH	0.1075	0.1783	0.2431	0.2650	0.547	0.555	0.656	0.735
SELVE	<u>0.2210</u>	<u>0.2915</u>	<u>0.3308</u>	<u>0.3677</u>	9.589	9.917	9.943	10.057
RSSH	0.2389	0.3117	0.3784	0.4167	18.589	19.107	19.842	23.120

Table 3: The MAP scores and training time of different methods on **Fashion-MNIST**. Note: the best results are in bold and the second-best are underlined.

Methods	Fashion-MNIST				Training time (Seconds)			
	16bits	32bits	64bits	96bits	16bits	32bits	64bits	96bits
LSH	0.2637	0.3119	0.3493	0.3916	0.001	0.012	0.020	0.038
PCAH	0.1934	0.2053	0.2285	0.2856	0.653	0.673	0.715	1.042
ITQ	0.4200	0.4350	0.4351	0.4482	2.064	3.910	6.478	9.948
DGH	<u>0.4364</u>	<u>0.4392</u>	<u>0.4467</u>	<u>0.4541</u>	80.983	89.739	99.222	140.074
SGH	0.3204	0.3240	0.3327	0.3412	4.700	16.691	44.283	71.048
SpH	0.3363	0.3530	0.3828	0.3922	9.796	15.159	24.469	33.740
SH	0.2860	0.2970	0.3037	0.3325	0.804	0.958	1.050	1.153
SELVE	0.2457	0.2746	0.3300	0.3781	18.051	18.849	19.325	19.664
RSSH	0.4925	0.4950	0.4954	0.5077	43.054	43.721	46.27	51.307

Table 4: The MAP scores and training time of different methods on **NUS-WIDE**. Note: the best results are in bold and the second-best are underlined.

Methods	NUS-WIDE				Training time (Seconds)			
	16bits	32bits	64bits	96bits	16bits	32bits	64bits	96bits
LSH	0.3481	0.3525	0.3585	0.3616	0.001	0.001	0.016	0.019
PCAH	0.3537	0.3569	0.3620	0.3678	0.934	0.968	0.972	1.009
ITQ	<u>0.3795</u>	<u>0.3836</u>	<u>0.3873</u>	<u>0.3911</u>	5.293	10.100	19.416	30.653
DGH	0.3389	0.3389	0.3619	0.3642	248.618	268.98	342.031	612.283
SGH	0.3418	0.3422	0.3432	0.3449	12.286	35.167	69.828	101.084
SpH	0.3706	0.3738	0.3773	0.3787	15.972	24.368	39.726	56.682
SH	0.3422	0.3721	0.3746	0.3752	1.272	1.425	1.867	2.281
SELVE	0.3589	0.3607	0.3681	0.3771	25.756	35.098	43.612	48.836
RSSH	0.3965	0.3982	0.4140	0.4173	322.742	433.324	778.636	996.642

which testifies the effectiveness of our proposed approach. By the way, we also collected the training time of all the competitors¹⁷; although RSSH is not the most efficient, yet the time cost is acceptable (even on the largest dataset NUS-WIDE, RSSH can finish learning to hashing within twenty minutes on a commonly configured PC). Therefore, the satisfactory image retrieval performance and fast efficiency reveal its potentials in real-world applications.

6.5. RSSH v.s. Deep Hashing Methods

Apart from the above competitions among shallow models, we further conducted experiments on **USPS**, **Caltech256** and **Fashion-MNIST**¹⁸ with two deep hashing methods, i.e., DeepBit [22] and SADH [21], as well as our RSSH. Here, in this section’s experiments, the color images of Caltech256 are cropped into 64×64 grayscale ones; then the inputs for these methods can be all the same with grayscale pixels. Specifically, the grayscale images are firstly scaled into the same sizes as the inputs of deep architec-

¹⁷Compared with the time spendings in the training stage, the time costs in the testing phase are quite small and could be ignored; thus, in this paper, we only recorded the training time for comparisons.

¹⁸Note that the NUS-WIDE dataset is not invited here because its original images are not available online.

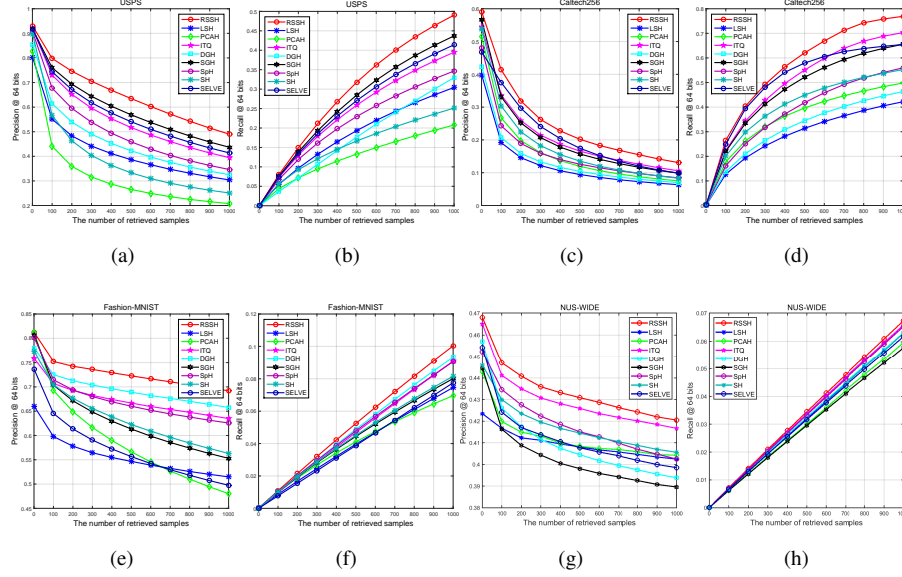


Figure 3: Precision and recall for the topN retrieved results on the selected datasets with 64 bits.

tures DeepBit¹⁵ and SADH¹⁶, and then they can be the direct inputs of these two deep hashings; while for RSSH, the grayscale images are vectorized as the inputs. In what follows, the MAP results are collected in Table 5.

Obviously, we could find that RSSH needs much less time than the other two deep approaches in learning to hash; this is quite natural that deep hashings are significantly complex with much more parameters as well as time-consuming iterative BP algorithms; while RSSH is a shallow model with fast designed iterative algorithms. What’s surprising is that RSSH yields higher MAP values than DeepBit and SADH on these image datasets. Noticeably, RSSH is not an end-to-end deep hashing model, but it shows much better performance than these state-of-the-art unsupervised deep hashing competitors, which further confirms the effectiveness of our multi-stage learning for hashing.

6.6. Parameter Sensitivity Analysis

There are several hyper-parameters, i.e., r , β , λ , η , σ and m designed in RSSH, and we conduct their sensitivity analysis on all the datasets with 64 bits. Particularly, we run a series of experiments by varying the value of one parameter while keeping the others

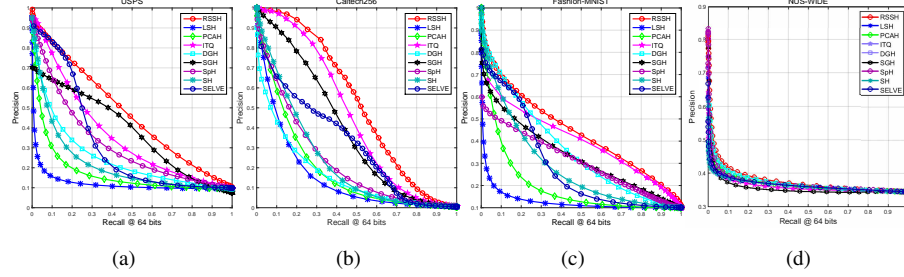


Figure 4: PR curves on the selected datasets with 64 bits.

Table 5: Competitions between RSSH and two deep hashing models on three benchmark datasets.

Datasets/Methods		16bits		32bits		64bits	
		MAP	Training time	MAP	Training time	MAP	Training time
USPS	DeepBit	0.2886	743.105	0.3215	913.565	0.3495	1024.374
	SADH	0.3109	1052.381	0.3896	1291.459	0.4215	1355.465
	RSSH	0.4940	8.848	0.4990	8.933	0.5073	9.488
Caltech256	DeepBit	0.1527	3360.172	0.2452	4896.909	0.2878	6942.376
	SADH	0.2187	4665.901	0.3094	6401.665	0.3576	8066.064
	RSSH	0.2332	115.269	0.3344	154.402	0.3781	229.232
Fashion-MNIST	DeepBit	0.4182	5398.629	0.4371	7550.568	0.4409	9490.672
	SADH	0.4309	8729.452	0.4649	9420.622	0.4687	11026.246
	RSSH	0.4925	43.054	0.4950	43.721	0.4954	46.270

fixed. Note that the benchmark parameters are ($r = 100$, $\beta = 0.1$, $\lambda = 1$, $\eta = 1e - 3$, $\sigma = 0.4$ and $m = 2,000$). Finally, the experimental results (MAP values) are collected and drawn in Fig. 5.

In regard to r , it's observed from Fig. 5a that the general trend of MAP values is to rise first and then fall down on the selected image collections. Recall that r controls the rank of the similarity matrix \mathbf{S} ; therefore, when r is small, the rank of \mathbf{S} is small but the loss of Eq. (3) would be large; however, when r is too large, the subspace structures of datasets couldn't be well revealed and the retrieval performance can be quite poor. Besides, the computational complexity of RSSH would go up when r rises. Hence, we finally set $r = 100$ for the experiments on all datasets due to the fact that it can contribute to not only competitive retrieval performance but also an acceptable

computational complexity.

Fig. 5b and Fig. 5c tell us that RSSH can achieve good performances with a large wide range of values w.r.t. parameters β and λ , and then we set them to 0.1 and 10 respectively. Seen from Fig. 5d, the MAP value at $\eta = 0$ is almost the same with those at $0 < \eta < 1$. This is because that there doesn't exist the phenomenon of overfitting and irreversibility in the selected datasets. However, we here set $\eta=1e-6$ instead of zero for generalizations to other datasets.

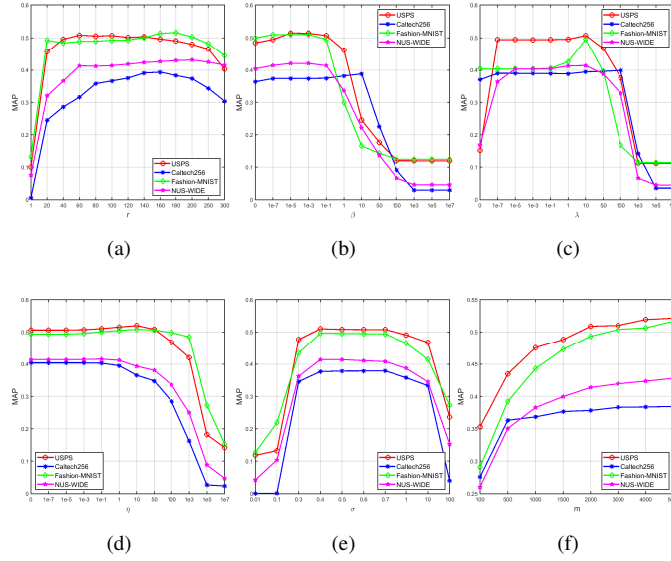


Figure 5: Parameter analysis (r , β , λ , η , σ and m) on different datasets with 64 bits.

Parameters σ and m correspond to the kernel width and the number of anchors respectively in the **RBF** kernel mapping: $\phi(\mathbf{x}) = [\exp(-\|\mathbf{x}-\mathbf{a}_1\|^2/(2\sigma^2)), \dots, \exp(-\|\mathbf{x}-\mathbf{a}_m\|^2/(2\sigma^2))]$. As can be seen from Fig. 5e, the MAP values on different image collections are with similar trends, i.e., RSSH yields good performances when σ is between 0.4 and 0.7. Thus, we set $\sigma = 0.5$ in this paper. From Fig. 5f, we can find that with the increasing of m , the MAP values are getting larger and larger on all the image collections. This is reasonable that it usually needs more basic vectors to well represent complex samples in large-scale dataset. However, the trend of growth is getting smaller and smaller with the increase of m , which probably tells us that we just need partial

samples as anchors for achieving satisfactory results. Hence, we configure m as 2,000 for competitive performance.

6.7. Convergence Investigation

To investigate the convergence of our designed Algorithm 2, we further plotted its loss curves with 64 bits¹⁹ on the four datasets in Fig. 6. Specifically, the y -axis is the normalized objective function value²⁰ and the x -axis denotes the iteration number. Clearly, we can see that Algorithm 2 can smoothly converge within 20 iterations, which is reasonably fast.

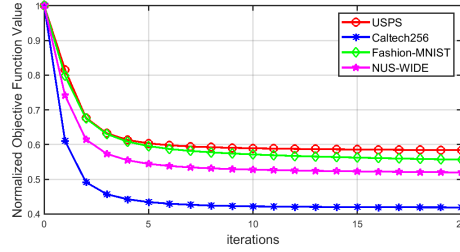


Figure 6: Convergence curves of RSSH on different datasets with 64 bits.

6.8. Ablation Study

Recall that our main characteristics are twofold: (1) LRR-based pairwise similarities which come from the intrinsic global structures of datasets; (2) a different optimization method which solves the classic challenging discrete graph hashing. Hence, RSSH is similar to DGH [15], which utilizes the anchor graph based pairwise similarities and its own optimization. With the aim to study the effectiveness of the above two properties, we replace the anchor-graph-based similarities with the LRR-based similarities in DGH and coin a new combination, termed as “LRR+DGH”. Besides, we could also interchange the anchor-graph-based similarities in DGH and the LRR-based similarities in RSSH with the original feature-based cosine similarities, dubbed “Cosine+DGH”

¹⁹The loss curves with other bits hold similar trends as Fig. 6.

²⁰Each iteration’s value is divided by the one at the first iteration.

Table 6: The effectiveness of LRR-based pairwise similarities and our optimizations for hashing.

Methods/Datasets	USPS				Caltech256			
	32bits		64bits		32bits		64bits	
	MAP	Training time	MAP	Training time	MAP	Training time	MAP	Training time
DGH	0.2652	22.569	0.2823	22.966	0.1282	38.910	0.1606	59.143
LRR+DGH	0.3841	21.029	0.4291	23.670	0.2150	82.567	0.2701	86.731
Cosine+DGH	0.2617	17.207	0.2776	21.393	0.1536	26.139	0.2172	38.747
Cosine+RSSH	0.3458	1.809	0.3671	3.372	0.2467	2.244	0.2738	4.693
RSSH	0.4990	8.933	0.5073	9.499	0.3117	19.107	0.3784	19.842

Methods/Datasets	Fashion-MNIST				NUS-WIDE			
	32bits		64bits		32bits		64bits	
	MAP	Training time	MAP	Training time	MAP	Training time	MAP	Training time
DGH	0.4392	89.739	0.4467	99.222	0.3389	268.980	0.3619	342.031
LRR+DGH	0.4715	126.661	0.4808	186.777	0.3447	603.118	0.3751	893.930
Cosine+DGH	0.3386	77.251	0.3748	91.097	0.3364	230.299	0.3407	314.016
Cosine+RSSH	0.3575	16.896	0.3902	27.878	0.3415	39.171	0.3689	65.772
RSSH	0.4950	43.721	0.4954	46.270	0.3982	433.324	0.4140	778.636

and “Cosine+RSSH” respectively. Further extensive experiments on the selected image datasets are conducted and the results are collected in Table 6.

Firstly, by comparing DGH and LRR+DGH, we can conclude that LRR-based similarities contribute more to the hashing retrieval than the anchor-based similarities because the MAP scores of LRR+DGH are always higher than those of DGH on all the image datasets. Similarly, by comparing Cosine+RSSH and RSSH, we could still find that RSSH performs much better than Cosine+RSSH in terms of MAP values, which actually indicates that LRR-based similarities contribute more to the hashing retrieval than the cosine similarities.

Secondly, taking the RSSH and LRR+DGH into considerations, we could achieve that our designed optimization approaches further drive up the retrieval performance against the optimization method in DGH. Likewise, taking the Cosine+DGH and Cosine+RSSH into account, the same result as above could also be summarized.

Finally, we could also discover that the designed algorithm in RSSH runs much faster than that in DGH by comparing the training time of Cosine+DGH and Cosine+RSSH, as well as LRR+DGH and RSSH. Here one may argue that Cosine+RSSH is the most efficient (also due to its low-rank representation of the similarity matrix) among these

adapted models, but the cosine similarities are inferior to the LRR-based similarities in achieving high retrieval precisions.

Overall, the well-designed experiments exhibit not only the effectiveness of LRR-based pairwise similarities, but also the further performance gain of our RSSH’s algorithm in image retrieval tasks.

7. A Retrieval Case on Caltech256

Fig. 8 shows the top-20 image retrieval results for three randomly chosen queries — T-shirt (Fig. 7a), airplane (Fig. 7b) and electric-guitar (Fig. 7c) — on Caltech256, using the above-mentioned competitive unsupervised hashing methods. The green bordered images indicate the relevant search results w.r.t. the queries; and the red bordered ones represent the opposite. Obviously, RSSH achieves the highest accuracies in regard to different queries, i.e., 20/20, 19/20 and 16/20 corresponding to T-shirt, airplane and electric guitar, which intuitively exhibits its clear advantages over other competitors.



(a) Query: T-shirt



(b) Query: airplane



(c) Query: electric guitar

Figure 7: Randomly selected image queries from Caltech256 dataset.

To be more specific, probably because T-shirt’s appearance is simple, several baseline methods (e.g., ITQ, SGH, SELVE) could also generate good retrieval results; however, they still sometimes make some errors, e.g., ITQ/SGH both mistake the 18-th image (indeed, they are somewhat similar in appearance with T-shirt) for the given query. For the other two more challenging queries, the differences between various approaches’ retrieved results are more sharp, e.g., methods like ITQ (the 17-th image in Fig. 8b), and SELVE (the 1st/4-th images in Fig. 8c) are more easily influenced by multi-objects in one image; nevertheless, RSSH could differentiate them because of its multi-subspace learning with LRR. In a word, the global-structure preserved similarities are beneficial



(a) Retrieval results: T-shirt



(b) Retrieval results: airplane



(c) Retrieval results: electric guitar

Figure 8: The retrieval results corresponding to the image queries in Fig. 7. Note that the bounding boxes are green for correct results and red for wrong ones.

for dealing with complex images, which makes RSSH competitive in hashing-based retrieval tasks.

8. Conclusion

This paper presents a novel unsupervised hashing method, RSSH for short. It firstly recovers the subspace structures of data by a new variant of the LRR model, and then constructs the similarity matrix in a time-and-space saving form of the learned correlation matrix. In what follows, RSSH solves the strongly binary-constrained graph hashing with the help of two surrogate variables, i.e., it converts the original discrete optimization problem into three subproblems with each addressed by a closed-form solution, which greatly render the whole learning converge quickly. Extensive competitions on four image datasets exhibit the obvious superiority of RSSH to quite a few state-of-the-art baseline hashing methods. Besides, what deserves further attention is that RSSH, a shallow model, defeated two recently proposed deep hashing models, which probably reveals that the potential of unsupervised deep hashing still remains to be tapped²¹. Specifically, the adversarial [48] and “self-supervised” [49, 50] deep hashing from unlabeled datasets are quite promising research directions.

In addition, although RSSH has shown the great benefits of global structure preserved similarities for hashing, yet it couldn’t handle streaming data samples which would be the future challenging work for more practical large-scale image retrieval systems with images increasing online.

9. Acknowledgment

This work is supported in part by the National Key Research and Development Program of China (under grant No. 2017YFB1400200) and China Postdoctoral Science Foundation. We also thank the Network Information Center of Beihang University for providing high-performance servers.

²¹As is generally believed that deep neural network based hashing approaches would perform better than shallow hashing methods in terms of image retrieval tasks.

References

- [1] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, VLDB (1999) 518–529.
- [2] F. Cakir, S. Sclaroff, Adaptive hashing for fast similarity search, ICCV (2015) 1044–1052.
- [3] Y. Guo, G. Ding, L. Liu, J. Han, L. Shao, Learning to hash with optimized anchor embedding for scalable retrieval, IEEE Trans. Image Processing 26 (2017) 1344–1354.
- [4] W. Liu, J. Wang, R. Ji, Y. Jiang, S. Chang, Supervised hashing with kernels, CVPR (2012) 2074–2081.
- [5] G. Lin, C. Shen, Q. Shi, A. van den Hengel, D. Suter, Fast supervised hashing with decision trees for high-dimensional data, CVPR (2014) 1971–1978.
- [6] F. Shen, C. Shen, W. Liu, H. T. Shen, Supervised discrete hashing, CVPR (2015) 37–45.
- [7] Q. Ma, C. Bai, J. Zhang, Z. Liu, S. Chen, Supervised learning based discrete hashing for image retrieval, Pattern Recognition 92 (2019) 156–164.
- [8] W. Kang, W. Li, Z. Zhou, Column sampling based discrete supervised hashing, AAAI (2016) 1230–1236.
- [9] X. Luo, L. Nie, X. He, Y. Wu, Z.-D. Chen, X.-S. Xu, Fast scalable supervised hashing, SIGIR (2018) 735–744.
- [10] Z. Zhang, G.-S. Xie, Y. Li, S. Li, Z. Huang, Sadih: Semantic-aware discrete hashing, AAAI (2019) 5853–5860.
- [11] J. Tang, Z. Li, X. Zhu, Supervised deep hashing for scalable face image retrieval, Pattern Recognition 75 (2018) 25–32.
- [12] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, NIPS (2009) 1753–1760.

- [13] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, NIPS (2009) 1042–1050.
- [14] W. Liu, J. Wang, S. Kumar, S. Chang, Hashing with graphs, ICML (2011) 1–8.
- [15] W. Liu, C. Mu, S. Kumar, S. Chang, Discrete graph hashing, NIPS (2014) 3419–3427.
- [16] Q. Jiang, W. Li, Scalable graph hashing with feature transformation, IJCAI (2015) 2248–2254.
- [17] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, NIPS (2001) 849–856.
- [18] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, ICML (2010) 663–670.
- [19] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 171–184.
- [20] Y. N. Li, P. Wang, Robust image hashing based on low-rank and sparse decomposition, ICASSP (2016) 2154–2158.
- [21] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, H. T. Shen, Unsupervised deep hashing with similarity-adaptive and discrete optimization, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2018) 3034–3044.
- [22] K. Lin, J. Lu, C. Chen, J. Zhou, Learning compact binary descriptors with unsupervised deep neural networks, CVPR (2016) 1183–1192.
- [23] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, Symposium on Computational Geometry (2004) 253–262.
- [24] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing for scalable image search, ICCV (2009) 2130–2137.

- [25] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, *IEEE Trans. Pattern Anal. Mach. Intell.* 35 (2013) 2916–2929.
- [26] V. E. Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, Deep hashing for compact binary codes learning, *CVPR* (2015) 2475–2483.
- [27] T. Do, A. Doan, N. Cheung, Learning to hash with binary deep neural network, *ECCV* (2016) 219–234.
- [28] E. Yang, C. Deng, T. Liu, W. Liu, D. Tao, Semantic structure-based unsupervised deep hashing, *IJCAI* (2018) 1064–1070.
- [29] C. Li, L. Lin, W. Zuo, W. Wang, J. Tang, An approach to streaming video segmentation with sub-optimal low-rank decomposition, *IEEE Trans. Image Processing* 25(5) (2016) 1947–1960.
- [30] S. R. Rao, R. Tron, R. Vidal, Y. Ma, Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories, *IEEE Trans. Pattern Anal. Mach. Intell.* 32 (2010) 1832–1845.
- [31] L. Chen, S. Chang, An adaptive learning algorithm for principal component analysis, *IEEE Trans. Neural Networks* 6 (1995) 1255–1263.
- [32] D. Keysers, H. Ney, Linear discriminant analysis and discriminative log-linear modeling, *ICPR* (2004) 156–159.
- [33] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, *Nature* 401 (1999) 788–791.
- [34] R. Vidal, Y. Ma, S. Sastry, Generalized principal component analysis (GPCA), *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (2005) 1945–1959.
- [35] E. Elhamifar, R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, *IEEE Trans. Pattern Anal. Mach. Intell.* 35 (2013) 2765–2781.

- [36] S. Li, Y. Fu, Learning robust and discriminative subspace with low-rank constraints, *IEEE Trans. Neural Netw. Learning Syst.* 27(11) (2016) 2160–2173.
- [37] T.-T. Do, A.-D. Doan, N.-M. Cheung, Learning to hash with binary deep neural network, *ECCV* (2016) 219–234.
- [38] C. Li, L. Lin, W. Zuo, J. Tang, M.-H. Yang, Visual tracking via dynamic graph learning, *IEEE trans. Pattern Anal. Mach. Intell.* (2018) 1–15.
- [39] X. Zhu, Z. Huang, H. Cheng, J. Cui, H. T. Shen, Sparse hashing for fast multimedia search, *ACM Trans. Inf. Syst.* 31 (2013) 9.
- [40] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset, California Institute of Technology.
- [41] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, *CoRR* (2017) 1–6.
- [42] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, NUS-WIDE: A real-world web image database from National University of Singapore, *CIVR* (2009) 1–9.
- [43] Z. Lin, G. Ding, M. Hu, J. Wang, Semantics-preserving hashing for cross-view retrieval, *CVPR* (2015) 3864–3872.
- [44] Z. Jin, Y. Hu, Y. Lin, D. Zhang, S. Lin, D. Cai, X. Li, Complementary projection hashing, *ICCV* (2013) 257–264.
- [45] P. Zhang, W. Zhang, W. Li, M. Guo, Supervised hashing with latent factor models, *SIGIR* (2014) 173–182.
- [46] J. Heo, Y. Lee, J. He, S. Chang, S. Yoon, Spherical hashing, *CVPR* (2012) 2957–2964.
- [47] X. Zhu, L. Zhang, Z. Huang, A sparse embedding and least variance encoding approach to hashing, *IEEE Trans. Image Processing* 23 (2014) 3737–3750.
- [48] K. Wang, Adversarial machine learning with double oracle, *IJCAI* (2019) 6472–6473.

- [49] X. Liu, J. van de Weijer, A. D. Bagdanov, Exploiting unlabeled data in cnns by self-supervised learning to rank, *IEEE Trans. Pattern Anal. Mach. Intell.* 41(8) (2019) 1862–1878.
- [50] Z. Feng, C. Xu, D. Tao, Self-supervised representation learning by rotation feature decoupling, *CVPR* (2019) 10364–10374.